UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/731,548	12/05/2003	Peter Szpak	MWS-101	4558

959 7590 02/23/2007
LAHIVE & COCKFIELD, LLP
ONE POST OFFICE SQUARE
BOSTON, MA 02109-2127

EXAMINER	
WATT, CHRIS A	
ART UNIT	**PAPER NUMBER**
2174	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	02/23/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No. 10/731,548	Applicant(s) SZPAK ET AL.
	Examiner Chris Watt	Art Unit 2174

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *05 December 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-39* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-39* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *05 December 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 31 and 33-39 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mantooth et al. ("Mantooth" US Patent No. 6236956) in view of Zink et al. ("Zink" US Patent No. 6738964).

Regarding independent claim 31, Mantooth teaches, in a graphical modeling environment (i.e. col. 2 lines 2-3 of Mantooth : " tools for configuring model-specific simulation controls"), a medium holding computer-executable instructions for a method, comprising the steps of: automatically changing parameters of the graphical model (i.e. col. 12 lines 31-33 of Mantooth : " A decision 510 permits the user to select either automatic or manual methods for modifying the sample points") that are inconsistent with the code generation specified by the user (i.e. col. 3 lines 64-67 of Mantooth : " Parameter Manager An interactive tool for inspecting and specifying valid operating regions, default values and descriptions of parameters in a model"). Mantooth does not teach displaying a user interface for prompting a user to specify one or more code generation goals

Zink teaches prompting a user (i.e. col. 14 lines 2-5 of Zink : ""wizards" that prompt the user for performance-related information and then process the user inputs to

determine actual property settings") to specify one or more code generation goals (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals"). It would have been obvious to an artisan at the time of the invention to combine the prompt of Zink with the graphical modeling environment of Mantooth "to provide the user with a powerful graphical design capability based on pre-engineered components" (col. 5 lines 30-32 of Zink)

Regarding independent claim 33, Mantooth teaches an apparatus comprising: at least one processor; a memory coupled to the at least one processor; and a computer program residing in the memory and being executed by the at least one processor, wherein the computer program includes a process for preparing a graphical model for a code generation process for creating code based on the graphical model (i.e. col. 6 line 32-36 of Mantooth : " Data comprising a model is made available to an analog hardware description language ("AHDL") code-generator 114 for generating AHDL code 116"). Mantooth does not teach a wizard for guiding a user through a process.

Zink teachers a wizard for guiding a user through a process (i.e. col. 14 lines 1-5 of Zink : "icon-based graphical user interfaces, dedicated property dialog windows, and via "wizards" that prompt the user for performance-related information and then process the user inputs to determine actual property settings"). It would have been obvious to an artisan at the time of the invention to combine the wizard of Zink with the graphical modeling environment of Mantooth "to provide the user with a powerful graphical design capability based on pre-engineered components" (col. 5 lines 30-32 of Zink)

Regarding dependent claim 34, see the analysis of claim 33 above. Mantooth, in combination with Zink teaches the apparatus of claim 33, wherein the wizard prompts the user (i.e. col. 14 lines 2-5 of Zink : ""wizards" that prompt the user for performance-related information and then process the user inputs to determine actual property settings") to specify (i.e. col. 23 lines 48-49 of Mantooth : " The user also is invited to specify specific model parameters or a range of parameters to sweep") at least one code generation goal (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals") for the embedded code (i.e. col. 6 lines 46-47 of Zink : " Each component includes some embedded build information").

Regarding dependent claim 35, see the analysis of claim 34 above. Mantooth, in combination with Zink teaches the apparatus of claim 34, wherein the wizard configures the graphical model (i.e. col. 2 lines 2-3 of Mantooth : " tools for configuring model-specific simulation controls") based on code generation goals specified by the user (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals").

Regarding dependent claim 36, see the analysis of claim 34 above. Mantooth, in combination with Zink teaches the apparatus of claim 34, wherein the computer program generates code in compliance with the code generation goals specified by the user (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals").

Regarding dependent claim 37, see the analysis of claim 34 above. Mantooth, in combination with Zink teaches the apparatus of claim 34, wherein the wizard prompts the user to select one or more conditions to be checked in the model (i.e. col. 14 lines 41-45 of Mantooth : " This graphical feedback, along with the indicated limiting

information, allows the user to select Newton steps for the model under selected circuit conditions, that will converge in a near optimal fashion").

Regarding dependent claim 38, see the analysis of claim 37 above. Mantooth, in combination with Zink teaches the apparatus of claim 37, wherein the wizard identifies objects in the model that do not comply with the selected conditions (i.e. col. 20 lines 47-49 of Zink : " Some components may have a "locking mechanism" that prevents normal users from modifying a component").

Regarding dependent claim 39, see the analysis of claim 37 above. Mantooth, in combination with Zink teaches the apparatus of claim 37, wherein the wizard modifies objects in the model that do not comply with the selected conditions (i.e. col. 20 lines 47-49 of Zink : " Some components may have a "locking mechanism" that prevents normal users from modifying a component").

3. Claims 1-30 and 32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mantooth et al. ("Mantooth" US Patent No. 6,236,956) in view of Zink et al. ("Zink" US Patent No. 6,738,964) and Iborra et al. ("Iborra" US Patent No. 7,137,100).

Regarding independent claim 1, Mantooth teaches a method for generating code from a model (i.e. col. 6 line 32-36 of Mantooth : " Data comprising a model is made available to an analog hardware description language ("AHDL") code-generator 114 for generating AHDL code 116") Mantooth does not teach that code is embedded, that generated code is in a compliable form, or prompting a user to specify at least one code generation goal for the embedded code.

Zink teaches embedded information (i.e. col. 6 lines 46-47 of Zink : " Each component includes some embedded build information"). It would have been obvious to an artisan at the time of the invention to combine the embedded information of Zink with the code generation method of Mantooth to permit "the expert (who creates a component) to embed a significant measure of his (or her) integration knowledge into the component" (col. 23 lines 55-60 of Zink). Zink further teaches the steps of prompting a user (i.e. col. 14 lines 2-5 of Zink : ""wizards" that prompt the user for performance-related information and then process the user inputs to determine actual property settings") to specify at least one code generation goal for the embedded code (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals"). Zink does not teach generating code in a compliable form.

Iborra teaches generating code in a compliable form (i.e. col. 6 lines 15-18 of Iborra : " Other species may start at the first statement and process it to make sure it complies with every applicable rule and then repeat this process for every other statement"). It would have been obvious to an artisan at the time of the invention to combine the compliable form of Iborra with the embedded information of Zink and the code generation model of Mantooth "to obtain a final software product which is compliant with the initial requirements, as represented in the source Conceptual Model" (col. 27 lines 46-48 of Iborra).

Regarding dependent claim 2, see the analysis of claim 1 above. Mantooth, in combination with Zink and Iborra, teaches the method of claim 1, further comprising the step of modifying (i.e. col. 8 line 52-55 of Mantooth : " Typically, what is referred to as

"model editing" actually involves modifying the values of previously-defined input parameters to make an existing model behave differently") one or more parameters of the graphical model (i.e. col. 10 lines 5-7 of Mantooth : " Changing a parameter definition, or introducing a new parameter, changes the model") to comply with the code generation goal (i.e. col. 6 lines 15-18 of Iborra : " Other species may start at the first statement and process it to make sure it complies with every applicable rule and then repeat this process for every other statement") in response to the user specifying said at least one code generation goal (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals").

Regarding dependent claim 3, see the analysis of claim 1 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 1, further comprising the step of providing feedback to the user (i.e. col. 3 lines 28-31 of Zink : " The software system includes a graphical user interface 404 for receiving user commands and to provide feedback (or results) to the user") regarding the compliance of the graphical model with a selected condition (i.e. col. 21 lines 33-35 of Zink : " For example a data viewer may provide an interactive window where the user can modify or simulate various operating conditions").

Regarding dependent claim 4, see the analysis of claim 3 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 3, wherein the user selects the selected condition through a user interface (i.e. col. 8 lines 12-14 of Mantooth : " A graphical user interface of the known type in which a mouse or other pointing device is used for selecting text or other display objects is preferred").

Regarding dependent claim 5, see the analysis of claim 4 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 4, wherein the user interface displays a list (i.e. col. 14 lines 6- of Zink : " Note that this property dialog window is composed of drop-down lists, labels, edit boxes, and buttons") of conditions to be checked (i.e. col. 9 lines 63-67 of Mantooth : "The parameter manager (described later) is used to define an acceptable range of values for this parameter, a default value, a descriptive comment, and the actual capacitance value to be used in a pre-simulation analysis such as model robustness checks"), and prompts the user to select one or more of the conditions (i.e. col. 14 lines 2-5 of Zink : ""wizards" that prompt the user for performance-related information and then process the user inputs to determine actual property settings").

Regarding dependent claim 6, see the analysis of claim 3 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 3, wherein the step of providing feedback to the user regarding the compliance of the graphical model with a selected condition comprises displaying a hyperlink (i.e. col. 6n lines 6-11 of Zink : " A hyperlink to an Internet website may be included in a component-token to assist users in getting additional information or in upgrading component-tokens to "full-strength" components") for linking the selected condition to an object of the graphical model that does not comply with the selected condition (i.e. col. 1 lines 24-27 of Zink : " This has led to the widespread employment of visual builder tools for creating custom application programs through the interlinking of software modules from libraries").

Regarding dependent claim 7, see the analysis of claim 3 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 3, further comprising the step of modifying an object of the model that does not comply with the selected condition (i.e. col. 20 lines 47-49 of Zink : " Some components may have a "locking mechanism" that prevents normal users from modifying a component").

Regarding dependent claim 8, see the analysis of claim 7 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 7, wherein the step of modifying comprises identifying the object and prompting the user to manually modify a parameter of the object (i.e. col. 12 lines 31-33 of Mantooth : " A decision 510 permits the user to select either automatic or manual methods for modifying the sample points").

Regarding dependent claim 9, see the analysis of claim 7 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 7, wherein the step of modifying comprises automatically modifying a parameter of the model to comply with the selected condition (i.e. col. 12 lines 31-33 of Mantooth : " A decision 510 permits the user to select either automatic or manual methods for modifying the sample points").

Regarding dependent claim 10, see the analysis of claim 1 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 1, wherein the graphical model is a block diagram (i.e. col. 4 lines 48-51 of Zink : " This enables the user to "draw" a block diagram (on a computer screen) comprised of distinct software functional units as well as distinct hardware functional units").

Regarding dependent claim 11, see the analysis of claim 1 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 1, wherein each code generation goal (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals") corresponds to a general code generation goal (i.e. col. 60 lines 31-36 of Iborra : " To sum up, the code automatically produced by the automatic software production system of one embodiment of the present invention corresponds to that of a true final software application, instead of that of just a prototype").

Regarding dependent claim 12, see the analysis of claim 11 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 11, further comprising the step of prompting the user to specify (i.e. col. 23 lines 48-49 of Mantooth : " The user also is invited to specify specific model parameters or a range of parameters to sweep") at least one detailed code generation goal (i.e. col. 3 lines 20-23 of Mantooth : " In the matrix viewer screen display, a selectable portion of a matrix is displayed, and displayed matrix cells can be selected to obtain more detailed information") for each specified general code generation goal (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals").

Regarding dependent claim 13, see the analysis of claim 12 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 12, further comprising the step of configuring the graphical model to comply with each detailed code generation goal (i.e. col. 2 lines 2-3 of Mantooth : " tools for configuring model-specific simulation controls").

Regarding independent claim 14, Mantooth teaches a method of preparing a graphical model for code generation (i.e. col. 6 line 32-36 of Mantooth : " Data comprising a model is made available to an analog hardware description language ("AHDL") code-generator 114 for generating AHDL code 116"), comprising the steps of displaying a user interface (i.e. col. 8 lines 12-14 of Mantooth : " A graphical user interface of the known type in which a mouse or other pointing device is used for selecting text or other display objects is preferred"); and automatically changing parameters of the graphical model (i.e. col. 12 lines 31-33 of Mantooth : " A decision 510 permits the user to select either automatic or manual methods for modifying the sample points") that are inconsistent specified by the user (i.e. col. 23 lines 48-49 of Mantooth : " The user also is invited to specify specific model parameters or a range of parameters to sweep"). Mantooth does not teach that the code is embedded, a user interface for prompting a user, or code generation goals.

Zink teaches embedded information (i.e. col. 6 lones 46-47 of Zink : " Each component includes some embedded build information"). It would have been obvious to an artisan at the time of the invention to combine the embeded information of Zink with the code generation method of Mantooth to permit "the expert (who creates a component) to embed a significant measure of his (or her) integration knowledge into the component" (col. 23 lines 55-60 of Zink). Zink further teaches the steps of prompting a user (i.e. col. 14 lines 2-5 of Zink : ""wizards" that prompt the user for performance-related information and then process the user inputs to determine actual

property settings") to specify at least one code generation goal for the embedded code (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals").

Regarding dependent claim 15, see the analysis of claim 14 above. Mantooth, in combination with Zink teaches the method of claim 14, further comprising the step of identifying a condition that does not comply with the code generation goals specified by the user (i.e. col. 3 lines 64-67 of Mantooth : " Parameter Manager An interactive tool for inspecting and specifying valid operating regions, default values and descriptions of parameters in a model").

Regarding independent claim 16, Mantooth teaches a method of preparing a model for code generation, the method comprising the steps of displaying a graphical user interface though which a user can specify code to be generated from the model (i.e. col. 6 line 32-36 of Mantooth : " Data comprising a model is made available to an analog hardware description language ("AHDL") code-generator 114 for generating AHDL code 116"). Mantooth does not teach that code is embedded, that generated code is in a compliable form, prompting a user to specify at least one code generation goal for the embedded code, or providing feedback regarding said code generation goal.

Zink teaches embedded information (i.e. col. 6 lones 46-47 of Zink : " Each component includes some embedded build information"). It would have been obvious to an artisan at the time of the invention to combine the embeded information of Zink with the code generation method of Mantooth to permit "the expert (who creates a component) to embed a significant measure of his (or her) integration knowledge into

the component" (col. 23 lines 55-60 of Zink). Zink further teaches the steps of prompting a user (i.e. col. 14 lines 2-5 of Zink : ""wizards" that prompt the user for performance-related information and then process the user inputs to determine actual property settings") to specify at least one code generation goal for the embedded code (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals") and providing feedback to the user regarding compliance of the graphical model with said code generation goal (i.e. col. 3 lines 28-31 of Zink : " The software system includes a graphical user interface 404 for receiving user commands and to provide feedback (or results) to the user"). Zink does not teach generating code in a compliable form.

Iborra teaches generating code in a compliable form (i.e. col. 6 lines 15-18 of Iborra : " Other species may start at the first statement and process it to make sure it complies with every applicable rule and then repeat this process for every other statement"). It would have been obvious to an artisan at the time of the invention to combine the compliable form of Iborra with the embedded information of Zink and the code generation model of Mantooth "to obtain a final software product which is compliant with the initial requirements, as represented in the source Conceptual Model" (col. 27 lines 46-48 of Iborra).

Regarding dependent claim 17, see the analysis of claim 16 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 16, further comprising the step of modifying one or more parameters of the graphical model to comply with the target characteristic in response to the user specifying said at least one target characteristic (i.e. col. 1 line 67-col. 2 line 2 of Mantooth : " In the area of model

robustness, intuitive tools are needed to perform model diagnostics that will help the modeler visualize both model characteristics and its interaction with the simulator").

Regarding dependent claim 18, see the analysis of claim 16 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 16, wherein the step of providing feedback comprises indicating to the user whether the graphical model complies with a selected condition (i.e. col. 6 lines 15-18 of Iborra : " Other species may start at the first statement and process it to make sure it complies with every applicable rule and then repeat this process for every other statement").

Claim 19 is similar in scope to claim 4, differing primarily in that claim 19 is directed towards a method of preparing a model and claim 4 is directed toward a method of generating embedded code, and is therefore rejected under similar rationale.

Claim 20 is similar in scope to claim 5, differing primarily in that claim 20 is directed towards a method of preparing a model and claim 5 is directed toward a method of generating embedded code, and is therefore rejected under similar rationale.

Claim 21 is similar in scope to claim 6, differing primarily in that claim 21 is directed towards a method of preparing a model and claim 6 is directed toward a method of generating embedded code, and is therefore rejected under similar rationale.

Claim 22 is similar in scope to claim 7, differing primarily in that claim 22 is directed towards a method of preparing a model and claim 7 is directed toward a method of generating embedded code, and is therefore rejected under similar rationale.

Claim 23 is similar in scope to claim 8, differing primarily in that claim 23 is directed towards a method of preparing a model and claim 8 is directed toward a method of generating embedded code, and is therefore rejected under similar rationale.

Claim 24 is similar in scope to claim 9, differing primarily in that claim 24 is directed towards a method of preparing a model and claim 9 is directed toward a method of generating embedded code, and is therefore rejected under similar rationale.

Claim 25 is similar in scope to claim 10, differing primarily in that claim 25 is directed towards a method of preparing a model and claim 10 is directed toward a method of generating embedded code, and is therefore rejected under similar rationale.

Regarding dependent claim 26, see the analysis of claim 16 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 16, further comprising the step of generating code that is compatible with said at least one target characteristic (i.e. col. 1 line 67-col. 2 line 2 of Mantooth : " In the area of model robustness, intuitive tools are needed to perform model diagnostics that will help the modeler visualize both model characteristics and its interaction with the simulator").

Regarding dependent claim 27, see the analysis of claim 16 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 16, wherein each target characteristic corresponds to a general code generation goal (i.e. col. 60 lines 31-36 of Iborra : " To sum up, the code automatically produced by the automatic software production system of one embodiment of the present invention corresponds to that of a true final software application, instead of that of just a prototype").

Regarding dependent claim 28, see the analysis of claim 27 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 27, further comprising the step of prompting the user to specify at least one detailed code generation goal (i.e. col. 3 lines 64-67 of Mantooth : " Parameter Manager An interactive tool for inspecting and specifying valid operating regions, default values and descriptions of parameters in a model") for each specified general code generation goal (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals").

Regarding dependent claim 29, see the analysis of claim 28 above. Mantooth, in combination with Zink and Iborra teaches the method of claim 28, further comprising the step of configuring the graphical model (i.e. col. 2 lines 2-3 of Mantooth : " tools for configuring model-specific simulation controls") to comply (i.e. col. 6 lines 15-18 of Iborra : " Other species may start at the first statement and process it to make sure it complies with every applicable rule and then repeat this process for every other statement") with each detailed code generation goal (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals").

Regarding independent claim 30, Mantooth teaches, in a graphical modeling environment, a medium holding computer-executable instructions for a method (i.e. col. 6 line 32-36 of Mantooth : " Data comprising a model is made available to an analog hardware description language ("AHDL") code-generator 114 for generating AHDL code 116"), comprising the steps of: displaying a graphical user interface though which a user can specify (i.e. col. 23 lines 48-49 of Mantooth : " The user also is invited to specify specific model parameters or a range of parameters to sweep") at least one

target characteristic for code to be generated from the graphical model (i.e. col. 1 line 67-col. 2 line 2 of Mantooth : " In the area of model robustness, intuitive tools are needed to perform model diagnostics that will help the modeler visualize both model characteristics and its interaction with the simulator"); and in response to a user specifying a target characteristic, providing feedback to the user regarding compliance of the graphical model with said target characteristic. Mantooth does not teach that code is embedded, that generated code is in a compliable form, or providing feedback regarding said code generation goal.

Zink teaches embedded information (i.e. col. 6 lines 46-47 of Zink : " Each component includes some embedded build information"). It would have been obvious to an artisan at the time of the invention to combine the embedded information of Zink with the code generation method of Mantooth to permit "the expert (who creates a component) to embed a significant measure of his (or her) integration knowledge into the component" (col. 23 lines 55-60 of Zink). Zink further teaches the steps of prompting a user (i.e. col. 14 lines 2-5 of Zink : ""wizards" that prompt the user for performance-related information and then process the user inputs to determine actual property settings") to specify at least one code generation goal for the embedded code (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals") and providing feedback to the user regarding compliance of the graphical model with said code generation goal (i.e. col. 3 lines 28-31 of Zink : " The software system includes a graphical user interface 404 for receiving user commands and to provide feedback (or results) to the user"). Zink does not teach generating code in a compliable form.

Iborra teaches generating code in a compliable form (i.e. col. 6 lines 15-18 of Iborra : " Other species may start at the first statement and process it to make sure it complies with every applicable rule and then repeat this process for every other statement"). It would have been obvious to an artisan at the time of the invention to combine the compliable form of Iborra with the embedded information of Zink and the code generation model of Mantooth "to obtain a final software product which is compliant with the initial requirements, as represented in the source Conceptual Model" (col. 27 lines 46-48 of Iborra).

Regarding independent claim 32, Mantooth teaches a method of generating code (i.e. col. 6 line 32-36 of Mantooth : " Data comprising a model is made available to an analog hardware description language ("AHDL") code-generator 114 for generating AHDL code 116").Mantooth does not teach that code is embedded, that generated code is in a compliable form, or providing feedback regarding said code generation goal.

Zink teaches embedded information (i.e. col. 6 lines 46-47 of Zink : " Each component includes some embedded build information"). It would have been obvious to an artisan at the time of the invention to combine the embedded information of Zink with the code generation method of Mantooth to permit "the expert (who creates a component) to embed a significant measure of his (or her) integration knowledge into the component" (col. 23 lines 55-60 of Zink). Zink further teaches the steps of prompting a user (i.e. col. 14 lines 2-5 of Zink : ""wizards" that prompt the user for performance-related information and then process the user inputs to determine actual

property settings") to specify at least one code generation goal for the embedded code (i.e. col. 25 lines 2-3 of Zink : "optimization based on user-defined goals") and providing feedback to the user regarding compliance of the graphical model with said code generation goal (i.e. col. 3 lines 28-31 of Zink : " The software system includes a graphical user interface 404 for receiving user commands and to provide feedback (or results) to the user"). Zink does not teach generating code in a compliable form.

Iborra teaches generating code in a compliable form (i.e. col. 6 lines 15-18 of Iborra : " Other species may start at the first statement and process it to make sure it complies with every applicable rule and then repeat this process for every other statement"). It would have been obvious to an artisan at the time of the invention to combine the compliable form of Iborra with the embedded information of Zink and the code generation model of Mantooth "to obtain a final software product which is compliant with the initial requirements, as represented in the source Conceptual Model" (col. 27 lines 46-48 of Iborra).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chris Watt whose telephone number is (571) 270-1046. The examiner can normally be reached on Monday-Thursday 6:30-4:00 Eastern.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kristine L. Kincaid can be reached on (571) 276-5619. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Chris A. Watt/

CAW

February 13, 2007

SY LUU
PRIMARY EXAMINER